MORSE CODE ON THE VZ200

A previous article described an adaptor to operate RTTY on the VZ200 computer. The adapter has now been modified to include Morse code.

Lloyd Butler VK5BF
18 Ottawa Avenue, Panorama, SA. 504

Expansion of the programme resident in the EPROM and minor changes to the wiring, have expanded the VZ200 RTTY adaptor to include encoding and decoding of Morse code. Morse speed can be varied over a range of approximately five to 35 words per minute. Resident messages, buffer storage and split screen operation, all used on RTTY, are also available for Morse operation.

HARDWARE CHANGES

To interface for Morse code, the 8251 USART functions DSR and DTR are used as one bit input and output ports respectively. DSR is simply wired in parallel with the existing data input (RXD). DTR is wired via a spare gate (V6-2), which is used to key the tone output from gate (U5-3). The circuit changes are illustrated in Figure 1.

For Morse code, the output tone is set at 2125Hz by the software and this can be used to feed the speech input of a transmitter. In a single side-band transmitter, CW transmission (A1) is generated and on a transmitter where carrier is not suppressed, MCM transmission (A2 or F2) is generated. Of course the latter is only permissible above 52MHz.

MORSE FORMAT

Morse format is based on the following:

Dash = three dots length
Space between dot or dash elements = one dot length
Space between characters = three dots length
Space between words = seven dots length

Speed is controlled by a selection code of one to eight and for the two lowest speeds (below 10 WPM), the spacing is increased to the following:
Space between characters = five dots length
Space between words = 13 dots length

There are a number of special Morse characters which are not available on the keyboard and not available as printed characters. These have been equated to available characters as follows:
Error = asterisk (*)
Double dash = dash (—)
Wait = plus (+)
Start of message = less than (<)
End of message = equals (=)
End of work = at (@)

Error is transmitted as six dots, instead of the standard eight, because six elements per Morse character is the maximum the system can process.

Morse characters are generated from a look-up table, one byte per character. Bits two to seven are used to store the individual element of a character, zero representing no element a dot and one representing a dash. Element are justified left, with the last element sen always in bit seven. The numeric value forme by this is added to the number of elements i the character and the sum is the value stored i the look-up table. For up to five elemer characters, it is an easy matter to extract th number of elements from bits zero to two an the dots and dashes elements from bits thre to seven. For six element characters, there i an overlap on bit 2 and summing causes b carry on four of these (parenthesis, comma colon, and semi-colon). To detect these is a bi tricky. The logic is to look for a one in either bit four or five and binary 010 in bits zero to two. I this logic is satisfied, the number of elements i assumed to be six and six is subtracted fron the byte value to obtain the element format ir bits two to seven.

Some examples of look-up table coding are shown in Figure 2.

OPERATION

Morse can be sent on line, direct from the keyboard and characters are encoded at the selected speed by the software. In this method of operation, character and word spacing are

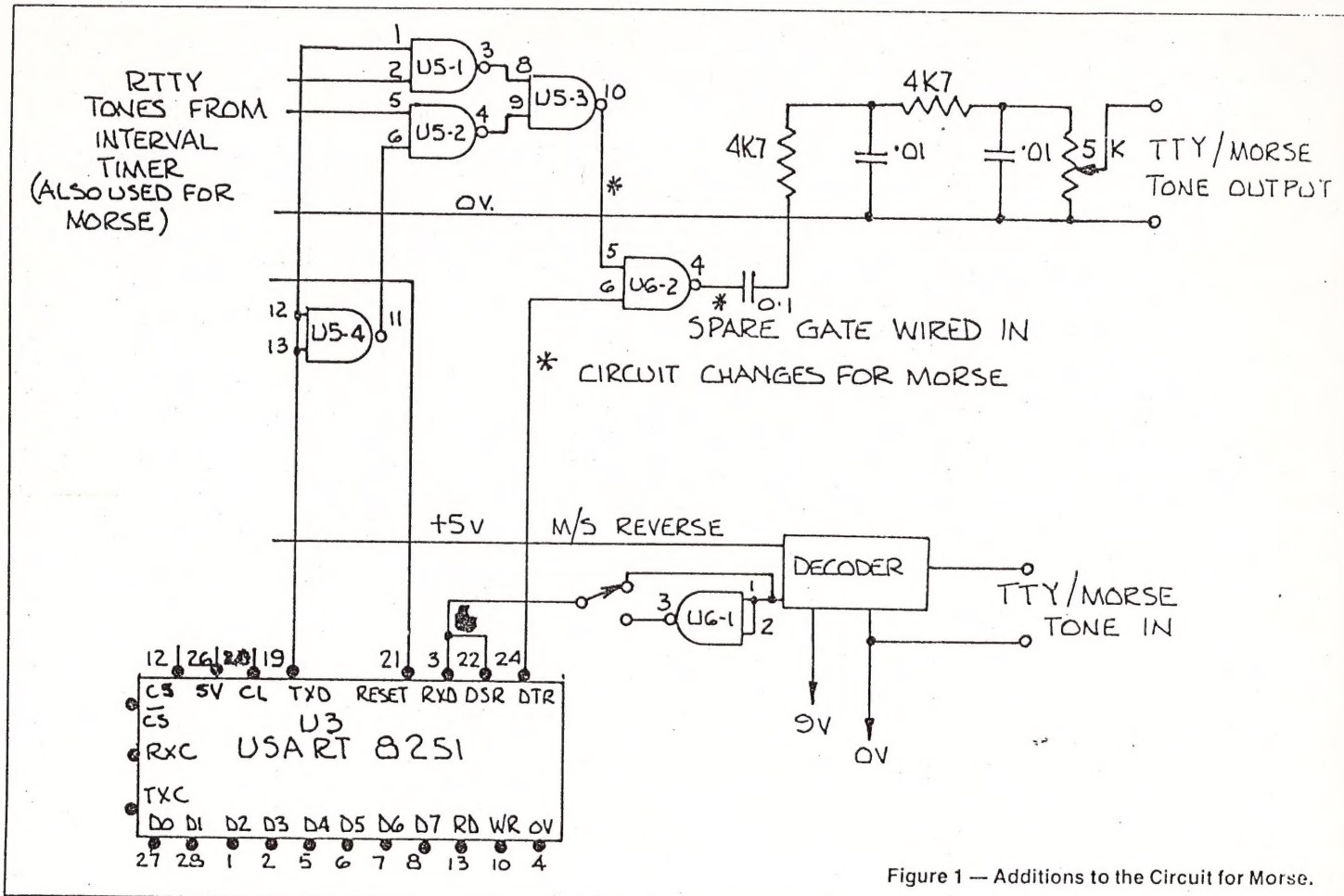

Figure 1 — Additions to the Circuit for Morse.

p. 19-20 1 of 2.

Figure 2 — Examples of Table Coding for Morse.

MORSE CODE	BINARY VALUE (BIT No)	HEX VALUE
	76543210	
Letter B — . . .	00010100	14
	code 4 elements	
Interrogation (?) . . — — . .	00110000	30
	code	
	+ 110	6
	6 elements	
	= 00110110	36
Comma (,) — — . . — —	11001100	CC
	code	
	+ 110	6
	6 elements	
	= 11010010	D2
	↗ ↗ ↗	
	Carry of Bit 2 into Bits 3 & 4	

determined by the time taken to move from one key to the next and, it seems to the writer, that a lot of practice would be needed to control the spacing correctly.

Morse is better sent by releasing the message from a pre-loaded buffer so that character and word spacing is accurately controlled by the computer. Using this method of operation, when communicating with another station, it is necessary to load the buffer at the same time as the other station is being received. This is common practice with RTTY operators using computers with split screen displays.

For RTTY, characters are encoded and decoded by the 8251 USART and the device is addressed by the computer for a very small proportion of the time. The rest of the time is available for other purposes including accessing the keyboard and loading the buffer, hence there is no problem in preparing the signal for transmission whilst the received signal is being decoded.

For Morse code, characters are encoded and decoded by timing loops called in by the main programme routine and while this is going on, access to the keyboard to load the buffer is denied. The obvious answer to the problem is to access the keyboard via an interrupt, however to make things difficult, the Z80 interrupt is already used by the VZ200 operating system. This calls an interrupt every 20 milliseconds on video vertical retrace.

Steve Onley described a method to make use of this 20 milli-second interrupt in Electronics Today International (ETI), May 1985. Your own interrupt is placed in series with that of the operating system so that it too can interrupt the main programme loop every 20 milli-seconds. The method described has been adopted for accessing the keyboard and loading the buffer in Morse operation.

Owing to peculiarities of the VZ200 system, keyboard access using this interrupt inhibits repetitive generation of a character, that is, you have to press the key each time a character is to be generated. This is not such a bad thing as it stops generation of more than one character if the key is accidentally pressed too long. The reason for the peculiarity is not clear as we do not have access to information on the VZ200 operating system.

The interrupt system works very well for loading the buffer, but a problem was found in attempting to generate Morse characters this way in real time. Because of the peculiarity discussed, a key pressed too soon, before the previous character is finished being transmitted, fails to generate a character and locks in this condition until the key is released and pressed again at the end of the previous character. Because of this problem, the interrupt is only used for loading the buffer and in all other modes of operation, the keyboard is accessed from the main programme loop. Using this method of access, the key can be kept pressed and the new character is sent, following a three dot length space, at the end of the previous character.

MEMORY

The combined RTTY and Morse programme package fully fills the 4k byte EPROM. A certain amount of programme trimming and rearrangement had to be carried out to fit it in. The programme is loaded in memory COO3H to CFF9H. RAM space used is 8000H to 8900H.

Based on information given by Jim Rowe in ETI, July 1985, the memory allocation should be suitable for both the VZ300 and VZ200 computers. A VZ300 has not been available to check it out, but the adaptor is expected to also work on the VZ300. There appears to be a change in clock frequency in the VZ300 from 3.580 to 3.540MHz. This will cause a shift in Baud rate and tone frequencies, but insufficient to be of significance.

CONCLUSION

The unit works very well on both RTTY and Morse code. The Morse decodes over a wide tolerance in reference to the speed selected. The writer was surprised how well it manages to decode hand sent Morse in which timing is not precisely defined. Noise interference is reduced by feeding the input signal via the RTTY decoder filters, but it does not perform as well as the human ear in separating Morse from noise. No doubt this could be improved if frequency shift keying were used.

Morse sent from the buffer sounds copperplate, as one would expect fully controlled by the computer. On line from the keyboard, the writer found it difficult to maintain constant character spacing, but this is probably a matter of practice on the keyboard. AR

2 of 2